What Is Claimed Is:

1. A program conversion apparatus for converting a given source program into a program for a multi-thread processor including a plurality of program counters and a plurality of thread execution apparatus, said plurality of thread execution apparatus being operable to fetch, decode and execute a plurality of instructions of threads simultaneously in accordance with said plurality of program counters such that it is possible to execute, after a thread is created, the thread in a control speculative mode wherein a change having had an effect on a register set can be cancelled later and to execute the thread in a data-dependent speculative mode wherein, when, after a self thread loads a value from a memory location, a parent thread by which the self thread has been created stores a value into the same memory location, at least a processing result of the self thread after the load is abandoned and the processing is re-executed, said multi-thread processor having an instruction set with which it can be executed by a single machine instruction or a combination of several machine instructions for a thread being executed by any of said thread execution apparatus to create a new thread of the control speculative mode, to end, if a designated condition is satisfied, the self thread and clear the control speculative mode of a thread of the control speculative mode created by the self thread, to abandon the created thread of the control speculative mode, to give, when a thread created by the self thread performs load from a memory

location of a designated address, an instruction in advance to temporarily block the operation, to clear the load temporary blocking instruction to the designated memory address, for the thread being executed by the thread execution apparatus to create a new thread of the data-dependent speculative mode and to clear the data-dependent speculative mode of the thread of the data-dependent speculative mode created by the self thread, said program conversion apparatus comprising:

a register allocation trial section for trying register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of an intermediate program;

a fork spot determination section for determining based on a result of the register allocation trial by said register allocation trial section whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used and determining a parallelization execution method with the parallel code;

an instruction reordering section for converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination by said fork spot determination section and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependence relationship between threads through a memory

into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage; and

a register allocation section for performing definite register allocation so that, regarding whether or not a physical register is allocated to the parallelized and reordered instruction sequence, the same allocation result as that upon the register allocation trial may be obtained.

2. A program conversion apparatus as claimed in claim 1, wherein said fork spot determination section investigates a data dependence relationship through a memory from a basic block in the intermediate program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block, counts, for each of the branching destinations, the instruction step number, from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence, and selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

3. A program conversion apparatus as claimed in claim 2, wherein said fork spot determination section determines the position of a data-dependent instruction through a memory in

each branching destination basic block using a value obtained by accumulating estimated execution cycle numbers of instructions in place of the instruction step number.

4. A program conversion apparatus as claimed in claim 1, wherein, upon conversion from a source program into a target program first, address coordination information for establishing coordination between the basic blocks of the intermediate program in said program conversion apparatus and machine language addresses of the target program to be outputted is outputted together with the target program, and a target program execution apparatus reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter, when said program conversion apparatus parallelizes the source program to convert the source program into a target program, said fork spot determination section refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly.

5. A program conversion apparatus as claimed in claim 4, wherein said fork spot determination section produces an

instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause date dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses, but investigates, when the number of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode but controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot.

6. A program conversion apparatus as claimed in claim 4, wherein said fork spot determination section investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block and synthesizes the investigated data dependence relationship and the conditional branching

probability obtained from the profile information, and if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.

7. A program conversion apparatus, comprising:

a syntax analysis section for analyzing the syntax of a source program to produce an intermediate program;

a parallelization section for performing optimization processing including parallelization for the intermediate program; and

a code generation section for producing a target program including an instruction code for a target processor apparatus from the intermediate program optimized by said parallelization section;

said parallelization section including an intermediate program inputting section for reading in the intermediate program and analyzing a control flow and a data flow, a register allocation section for trying to perform register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program and executing allocation of registers,

a fork spot determination section for determining, based on a result of the trial of the register allocation, a spot of a conditional branch portion of the intermediate program to be converted into a parallel code for which a thread creation instruction is used, an instruction reordering section for performing reordering of instructions before and after the parallelization spot from information of the parallelization spot determined by said fork spot determination section, the data flow and so forth, and an intermediate program outputting section for outputting the instruction sequence for which the conversion including the parallelization has been completed in a format of the intermediate program again.

8. A program conversion apparatus as claimed in claim 7, wherein said parallelization section includes a profile information inputting section for receiving profile information outputted from the target processor apparatus as a result of execution of the target program and converting the profile information into information of an internal format, and said fork spot determination section determines, based on the result of the register allocation trial and the profile information, a spot of a conditional branch portion of the intermediate program to be converted into a parallel code in which the thread generation code is used and determines a parallelization execution method by the parallel code.

9. A program conversion apparatus as claimed in claim 7, wherein said instruction reordering section converts

conditional branching portions in the intermediate program into a parallel code in which the thread creation instruction is used based on a result of the determination by said fork spot determination section, refers to the result of the register allocation trial to insert an instruction for assuring a data dependence relationship between threads through a memory into positions before and after the thread creation instruction and reorders the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage.

10. A program conversion apparatus as claimed in claim 7, wherein said register allocation section performs definite register allocation so that, regarding whether or not a physical register is allocated to the parallelized and reordered instruction sequence, the same allocation result as that upon the register allocation trial may be obtained.

11. A program conversion apparatus for performing optimization processing including parallelization of an intermediate program obtained by a syntax analysis of a source program performed by a syntax analysis section so that the intermediate program may be suitable for a target processor apparatus, comprising:

register allocation trial means for trying allocation of registers of the target processor apparatus on the intermediate program and obtaining register allocation information prior to actual allocation;

means for calculating a distance of data dependence generated through a memory in the target processor apparatus for the intermediate program;

means for determining a fork designation taking the distance of data dependence through a memory on the intermediate program into consideration and replacing a conditional branch with a thread creation instruction; and

means for referring to a result of the register allocation trial to reorder the instructions before and after the thread creation instruction on the intermediate program.

12. A program conversion apparatus according to claim 11, wherein said means for replacing a conditional branch with a thread creation instruction includes:

means for calculating a minimum value of data dependence of intermediate terms and variables for each of the two branching destinations of the conditional branch; and

means for comparing the two minimum values of the distance of data dependence determined for the two branches of the conditional branch, determining, when the two minimum values have a difference greater than a predetermined value, the branching direction of the branch which exhibits the higher minimum value of the distance of data dependence as a fork destination and selecting the conditional branch spot as a fork spot, but determining, when the two minimum values of the distance of data dependence do not have a difference equal to or greater than the predetermined value, that branching destination which

has been a branching destination in the original intermediate program as a fork destination and selecting the conditional branch spot as a fork spot candidate.

13. A program conversion apparatus as claimed in claim 11, further comprising:

means for receiving profile information outputted from a processor apparatus which executes the target program outputted from said program conversion apparatus and calculating a conditional branching probability and a data dependence occurrence frequency from the profile information; and

means for determining a fork destination and a data dependence assurance system from the distance of data dependence, the conditional branch probability and the data dependence occurrence frequency, and the number of spots of different memory addresses which cause data dependence and replacing the conditional branch with the thread creation instruction.

14. A program conversion apparatus as claimed in claim 7, wherein the target processor apparatus is a multi-thread processor which includes a plurality of program counters and a plurality of thread execution apparatus, said plurality of thread execution apparatus being operable to fetch, decode and execute a plurality of instructions of threads simultaneously in accordance with said plurality of program counters such that it is possible to execute, after a thread is created, the thread in a control speculative mode wherein a change having had an

effect on a register set can be canceled later and to execute the thread in a data-dependent speculative mode wherein, when, after a self thread loads a value from a memory location, a parent thread by which the self thread has been created stores a value into the same memory location, at least a processing result of the self thread after the load is abandoned and the processing is re-executed, said multi-thread processor having an instruction set with which it can be executed by a single machine instruction or a combination of several machine instructions for a thread being executed by any of said thread execution apparatus to create a new thread of the control speculative mode, to end, if a designated condition is satisfied, the self thread and clear the control speculative mode of a thread of the control speculative mode created by the self thread, to abandon the created thread of the control speculative mode, to give, when a thread created by the self thread performs load from a memory location of a designated address, an instruction in advance to temporarily block the operation, to clear the load temporary blocking instruction to the designated memory address, for the thread being executed by the thread execution apparatus to create a new thread of the data-dependent speculative mode and to clear the data-dependent speculative mode of the thread of the data-dependent speculative mode created by the self thread.

15. A program conversion method for performing an optimization process including parallelization for an

intermediate program outputted as a result of a syntax analysis on a program conversion apparatus which compiles a source program and outputs a target program for a target processing apparatus of the multi-thread type, comprising:

a register allocation trial step of trying register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program;

a fork spot determination step of determining based on a result of the register allocation trial whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used or performing determination of whether or not the conditional branch portion should be converted into a parallel code and, when such conversion should be performed, determination of a parallelization execution method;

an instruction reordering step of converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination by the fork spot determination step and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependence relationship between threads through a memory into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early

stage; and

a register allocation step of performing definite register allocation so that the same allocation result as that upon the register allocation trial may be obtained for the parallelized and reordered instruction sequence.

16. A program conversion method as claimed in claim 15, wherein the fork spot determination step means calculates a minimum value of data dependence of intermediate terms and variables for each of the two branching destinations of the conditional branch, compares the two minimum values of the distance of data dependence determined for the two branches of the conditional branch, determines, when the two minimum values have a difference greater than a predetermined value, the branching direction of the branch which exhibits the higher minimum value of the distance of data dependence as a fork destination and selects the conditional branch spot as a fork spot, but determines, when the two minimum values of the distance of data dependence do not have a difference equal to or greater than the predetermined value, that branching destination which has been a branching destination in the original intermediate program as a fork destination and selects the conditional branch spot as a fork spot candidate.

17. A program conversion method as claimed in claim 15, wherein the fork spot determination step investigates a data dependence relationship through a memory from a basic block having no branch and no confluence from within the intermediate

program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block, counts, for each of the branching destinations, the instruction step number, from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence, and selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

18. A program conversion method as claimed in claim 17, wherein the fork spot determination step determines the position of a data-dependent instruction through a memory in each branching destination basic block using a value obtained by accumulating estimated execution cycle numbers of instructions in place of the instruction step number.

19. A program conversion method as claimed in claim 15, wherein, upon conversion from a source program into a target program first by said program conversion apparatus, address coordination information for establishing coordination between the basic blocks of the intermediate program and machine language addresses of the target program to be outputted is outputted together with the target program, and a processor apparatus which is to execute the object program reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch

profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter, when said program conversion apparatus parallelizes the source program to convert the source program into a target program, the fork spot determination step refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly.

20. A program conversion method as claimed in claim 19, wherein the fork spot determination step produces an instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause date dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses, but investigates, when the number of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined

probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode but controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot.

21. A program conversion apparatus as claimed in claim 19, wherein the fork spot determination step investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block and synthesizes the investigated data dependence relationship and the conditional branching probability obtained from the profile information, and if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.

22. A program conversion method as claimed in claim 15, wherein the fork spot determination step includes the steps of:

discriminating whether or not the conditional branching

instruction corresponds to a return branch of a loop structure in the intermediate program;

determining, when the conditional branching instruction corresponds to a loop return branch, the direction of the return branch, which is a loop continuing direction, as a fork destination and selecting the conditional branch spot as a fork spot;

but calculating, when the conditional branching instruction is not a loop return branch, a minimum value of the distance of data dependence of intermediate terms/variables for each of the two branch destinations of the conditional branch;

comparing the two minimum values of the distance of data dependence determined with regard to the two branches of the conditional branch with each other to discriminate whether or not the two minimum values have a difference greater than a predetermined value; and

determining, when the two minimum values of the distance of data dependence have a difference greater than the predetermined value, the branch which exhibits a larger one of the minimum values of the distance of data dependence as a fork destination and selecting the conditional branching spot as a fork spot; but

determining, when the two minimum values of the distance of data dependence do not have a difference greater than the predetermined value, that one of the branches which has been a branch destination in the original intermediate program as

a fork destination and selecting the conditional branching spot as a fork candidate.

23. A program conversion method as claimed in claim 22, wherein the distance of data dependence is the number of steps in the intermediate program which represents at what distance from the top of a basic program of the branching destination the memory reference instruction is positioned with regard to each of the intermediate terms and variables which are defined in the basic block which is a processing object at present and may possibly be referred to in the branching destination and besides are estimated to be arranged on the memory.

24. A program conversion method as claimed in claim 22, wherein, upon the determination of the distance of data dependence, the number of cycles estimated to be required when pertaining instructions are executed on a processor of an object architecture.

25. A program conversion method as claimed in claim 15, wherein the instruction reordering step includes:

a first step of investigating an allocation situation of registers with regard to whether each of intermediate terms and variables in the intermediate program is coordinated with a register or a memory;

a second step of replacing a branching instruction positioned at the tail end of a basic instruction which is a processing object at present with a control speculation mode FORK instruction while the fork destination which is an operand

of the control speculation mode FORK instruction is determined as the fork destination selected by the fork spot determination step;

a third step of moving a branching condition expression positioned immediately prior to the control speculation mode FORK instruction in the intermediate instruction to the position immediately next to the control speculation mode FORK instruction and inserting to the tail end of the basic block, which is the destination of the movement of the branching condition expression, an instruction sequence for ending, when the branching condition is satisfied, the self thread and placing a child thread into a settlement mode which is a non-control speculation mode, but abandoning, when the branching condition is not satisfied, the child thread and keeping the self thread to continue execution of a succeeding instruction train;

a fourth step of moving each of instruction statements which are on the upstream side with respect to the control speculation mode FORK instruction in the basic block being a processing object at present and are to be substituted into intermediate terms and variables coordinated with a memory to a position on the downstream side with respect to the control speculation FORK instruction and inserting a BLOCK setting instruction to the position immediately prior to the control speculation mode FORK instruction while inserting a BLOCK clear instruction to the position immediately next to the movement destination of the substitute statement; and

a fifth step of issuing an instruction to allocate the registers in accordance with the register allocation situation assumed by the fork conversion processing in the second step.

26. A program conversion method as claimed in claim 15, wherein the fork spot determination step includes:

a first step of discriminating whether or not the conditional branching instruction corresponds to a return branch of a loop structure in the intermediate program;

a second step of provisionally determining, when the conditional branching instruction corresponds to a return branch of a loop structure, the direction of the return branch as a fork destination;

a third step of calculating, based on the received profile information, a probability with which a taken side of the conditional branching instruction is selected and another probability with which a fall-through side of the conditional branching instruction is selected;

a fourth step of discriminating whether or not the two calculated probabilities of the branches have a difference greater than a predetermined value;

a fifth step of provisionally determining, if the difference between the two probabilities of the branches is greater than the predetermined value, the branch which exhibits the higher probability as a fork destination;

a sixth step of calculating a minimum value of the distance of data dependence for each of the two branching destinations

of the conditional branch;

a seventh step of comparing the two minimum values of the distance of data dependence determined with regard to the two branches of the conditional branch with each other to discriminate whether or not the two minimum values have a difference greater than a predetermined value;

an eighth step of determining, when the two minimum values of the distance of data dependence have a difference greater than the predetermined value or no data dependence is found, that one of the branches which exhibits a higher one of the minimum values of the distance of data dependence as a fork destination;

a ninth step of calculating a minimum value of the distance of data dependence for the fork destination provisionally determined in the second step or the fifth step and discriminating whether or not the minimum value of the distance of data dependence of the provisionally determined fork destination is equal to or higher than a predetermined value;

a tenth step of settling, if the minimum value of the distance of data dependence of the provisionally determined fork destination is equal to or higher than a predetermined value or no data dependence through a memory is found, the fork destination provisionally determined in the second step or the fifth step as a formal fork destination;

an eleventh step of excepting, when it is determined that the minimum value of distance of data dependence is lower than

the predetermined value, the basic block from a fork spot;

a twelfth step of calculating a data dependence occurrence frequency from the received profile information;

a thirteenth step of discriminating whether or not the data dependence occurrence frequency is equal to or higher than a fixed level, counting, when the data dependence occurrence frequency is lower than the fixed level, the number of intermediate terms/variables on the memory which may possibly cause data dependence from the basic block of the fork source to the basic block of the fork destination in the intermediate program, discriminating whether or not the count value representing a data-dependent spot number is equal to or higher than a fixed level, determining that a fork according the DSP system is used if the data-dependent spot number is equal to or higher than the fixed level but determining that a fork according to the BLOCK system is used if the data-dependent spot number is lower than the fixed level, and providing information of the fork to the FORK instruction in the intermediate program; and

a fourteenth step of counting, when the data dependence occurrence frequency is equal to or higher than the fixed level, the number of data-dependent variables on the memory and determining that a fork according to the BLOCK system is used if the counted number is smaller than a fixed level but removing the basic block from a fork candidate if the counted number is equal to or greater than the fixed level.

27. A program conversion method as claimed in claim 15, wherein the instruction reordering step includes:

a first step of investigating an allocation situation of registers with regard to whether each of intermediate terms and variables in the intermediate program is coordinated with a register or a memory;

a second step of replacing a branching instruction positioned at the tail end of a basic instruction which is a processing object at present with a control speculation mode FORK instruction while the fork destination which is an operand of the control speculation mode FORK instruction is determined as the fork destination selected by the fork spot determination step;

a third step of moving a branching condition expression positioned immediately prior to the control speculation FORK instruction in the intermediate instruction to the position immediately next to the control speculation FORK instruction and inserting to the tail end of the basic block, which is the destination of the movement of the branching condition expression, an instruction sequence for ending, when the branching condition is satisfied, the self thread and placing a child thread into a settlement mode which is a non-control speculation mode, but abandoning, when the branching condition is not satisfied, the child thread and keeping the self thread to continue execution of a succeeding instruction train;

a forth step of checking whether a fork data assurance

system of the fork spot determined by the fork spot determination step is the BLOCK system or the DSP system;

a fifth step of moving, when the fork data assurance system is the BLOCK system, a memory store statement prior to the fork to a position after the fork, inserting necessary BLOCK setting and BLOCK clear instructions, inspecting, upon the movement, a data dependence relationship and moving only those instructions a change of whose instruction execution order does not change a result of arithmetic operation;

a sixth step of modifying, when the fork data assurance system is the DSP system, the FORK instruction produced by replacement in the second step so that a substitute statement into an intermediate term coordinated with a memory is moved to a position next to the FORK instruction to perform the fork in a data-dependent speculation mode; and

a seventh step of issuing an instruction to allocate the registers in accordance with the register allocation situation assumed by the fork conversion process in the second step.

28. A recording medium on which a program for causing a computer to perform an optimization process including parallelization for an intermediate program outputted as a result of a syntax analysis on a compiler which compiles a source program and produces and outputs a target program for a multi-thread processor apparatus is recorded, the optimization process including:

a register allocation trial process of trying register

allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program;

a fork spot determination process of determining based on a result of the register allocation trial whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used or performing determination of whether or not the conditional branch portion should be converted into a parallel code and, when such conversion should be performed, determination of a parallelization execution method;

an instruction reordering process of converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination by the fork spot determination step and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependent relationship between threads through a memory into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage; and

a register allocation process of performing definite register allocation so that the same allocation result as that upon the register allocation trial with regard to whether a physical register is allocated may be obtained for the

parallelized and reordered instruction sequence.

29. A recording medium as claimed in claim 28, wherein the fork spot determination process investigates a data dependence relationship through a memory from a basic block in the intermediate program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block, counts, for each of the branching destinations, the instruction step number, from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence, and selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

30. A medium as claimed in claim 17, wherein the fork spot determination process determines the position of a data-dependent instruction through a memory in each branching destination basic block using a value obtained by accumulating estimated execution cycle numbers of instructions in place of the instruction process number.

31. A medium as claimed in claim 28, wherein, upon conversion from a source program into a target program first by said compiler, address coordination information for establishing coordination between the basic blocks of the intermediate program and machine language addresses of the target program to be outputted is outputted together with the

target program, and a processor apparatus which is to execute the object program reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter, when said compiler parallelizes the source program to convert the source program into a target program, the fork spot determination process refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly.

32. A medium as claimed in claim 31, wherein the fork spot determination process produces an instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause date dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses, but investigates, when the number

of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode but controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot.

33. A compiler as claimed in claim 31, wherein the fork spot determination process investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block and synthesizes the investigated data dependence relationship and the conditional branching probability obtained from the profile information, and if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.